# PSet4 Section 5 Q1

AUTHOR
Charisma Lambert Prashanthi Subbiah

PUBLISHED
November 10, 2024

**Due 11/9 at 5:00PM Central. Worth 100 points + 10 points extra credit.**

## Submission Steps (10 pts)

1. This problem set is a paired problem set.
2. Play paper, scissors, rock to determine who goes first. Call that person *Partner 1*.
    - Partner 1 (name and cnet ID): Prashanthi Subbiah (prashanthis)
    - Partner 2 (name and cnet ID): Charisma Lambert (charisml)
3. Partner 1 will accept the `ps5` and then share the link it creates with their partner. You can only share it with one partner so you will not be able to change it after your partner has accepted.
4. "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **CL** **PS**
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" (1 point)
6. Late coins used this pset: **\0\2** Late coins left after submission: **\0\4** (1 used per partner, 2 left for each partner)
7. Knit your `ps5.qmd` to an PDF file to make `ps5.pdf`,
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
8. (Partner 1): push `ps5.qmd` and `ps5.pdf` to your github repo.
9. (Partner 1): submit `ps5.pdf` via Gradescope. Add your partner on Gradescope.
10. (Partner 1): tag your submission in Gradescope

```
import pandas as pd
import altair as alt
import time

import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

```
RendererRegistry.enable('png')
```

## Step 1: Develop initial scraper and crawler

### 1. Scraping (PARTNER 1)

```
import pandas as pd
import requests
from bs4 import BeautifulSoup

url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)
```

```
soup = BeautifulSoup(response.content, 'lxml')

#print(response)
#print(soup)

main_nochildren = soup.find_all(lambda t: t.name == 'main' and not
t.find_all())

# Find all <h2> tags within <main>
h2_tags = soup.find_all('h2')

# Extract links from <a> tags within each <h2>
links = []
for h2 in h2_tags:
    a_tag = h2.find('a', href=True)
    if a_tag:
        links.append(a_tag['href'])

# Print the extracted links
#print(links)

# Extracting titles of offences corresponding to each link
links_text = []
for h2 in h2_tags:
    a_tag = h2.find('a', href=True)
    if a_tag:
        links_text.append(a_tag.text)

#print(links_text) - commented out after checking

# Extracting Dates
# Find all <header> elements
header_tags = soup.find_all('header')

# Extract text from <span> tags within each <div> inside <header>
span_texts = []
for header in header_tags:
    div_tags = header.find_all('div')
    for div in div_tags:
        span_tag = div.find('span')
        if span_tag:
            span_texts.append(span_tag.text.strip())

# Print the extracted text
filtered_texts = span_texts[4:]

#print(filtered_texts) - commented out after checking

# Filtering for Categories
# Find all <header> elements
header_tags = soup.find_all('header')

# Extract text from <li> tags within each <ul> inside <div> within <header>
li_texts = []
```

```
for header in header_tags:
    div_tags = header.find_all('div')
    for div in div_tags:
        ul_tags = div.find_all('ul')
        for ul in ul_tags:
            li_tags = ul.find_all('li')
            for li in li_tags:
                li_texts.append(li.text.strip())

# Assuming you have a list called `li_texts`
# Get the last 20 elements
last_20_texts  =  li_texts[-20:]

#print(last_20_texts) - commented out after checking

# Create a DataFrame from the lists
df = pd.DataFrame({
    'Title': links_text,
    'Date': filtered_texts,
    'Category': last_20_texts,
    'Link': links
})

# Print the DataFrame
print(df.head(10))

# BingChat Query: How do I filter for an <a> tag in an <h2> tag?
```

```
                                               Title              Date  \
0  Pharmacist and Brother Convicted of $15M Medic...  November 8, 2024
1  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
2  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
3  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
4  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 2024
5  Former Licensed Counselor Sentenced For Defrau...  November 6, 2024
6  Macomb County Doctor And Pharmacist Agree To P...  November 4, 2024
7  Rocky Hill Pharmacy And Its Owners Indicted Fo...  November 4, 2024
8  North Texas Medical Center Pays $14.2 Million ...  November 4, 2024
9  New England Doctor Pleads Guilty To Drug Distr...  November 4, 2024

                    Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions
5  Criminal and Civil Actions
6  Criminal and Civil Actions
7  Criminal and Civil Actions
8  Criminal and Civil Actions
9  Criminal and Civil Actions

                                                Link
0  /fraud/enforcement/pharmacist-and-brother-conv...
```

```
/fraud/enforcement/boise-nurse-practitioner-se...
/fraud/enforcement/former-traveling-nurse-plea...
/fraud/enforcement/former-arlington-resident-s...
/fraud/enforcement/paroled-felon-sentenced-to-...
/fraud/enforcement/former-licensed-counselor-s...
/fraud/enforcement/macomb-county-doctor-and-ph...
/fraud/enforcement/rocky-hill-pharmacy-and-its...
/fraud/enforcement/north-texas-medical-center-...
/fraud/enforcement/new-england-doctor-pleads-g...
```

## 2. Crawling (PARTNER 1)

```
base_url = 'https://oig.hhs.gov'

# This extracts all text from the second <li> tag
agency_info_list = [] # Add your list of URLs here
for i in df['Link']:
    # Specify the URL
    url = base_url + i

    # Fetch the page content
    response = requests.get(url)
    response.raise_for_status()

    # Parse the HTML content with Beautiful Soup
    soup = BeautifulSoup(response.text, 'html.parser')

    # Use the CSS selector path to find the element
    agency_info = soup.select_one('#main-content > div > div:nth-child(2) > article > div >
        ul > li:nth-child(2)')

    # Check if the element is found and print its text
    if agency_info:
        agency_info_list.append(agency_info.get_text(strip=True))
    else:
        agency_info_list.append("Agency information not found.")

# Taking out word "Agency:"
cleaned_agency_info_list = [info.replace("Agency:", "").strip() for info in
        agency_info_list]

# Appending 'Enforcement Agency' column to df
df['Enforcement Agency'] = cleaned_agency_info_list
df.head(10)

# BingChat Query: How do I filter for a <li> tag, which has a nested <span> tag that has the
        text "Agency:" in it?

# Here is the XPath: '#main-content > div > div:nth-child(2) > article > div > ul > li:nth-
        child(2)'
```

| | Title | Date | Category | Link | Enforcement Agency |
|---|---|---|---|---|---|
| 0 | Pharmacist and Brother Convicted of $15M Medic... | November 8, 2024 | Criminal and Civil Actions | /fraud/enforcement/pharmacist-and-brother-conv... | U.S. Department of Justice |
| 1 | Boise Nurse Practitioner Sentenced To 48 Month... | November 7, 2024 | Criminal and Civil Actions | /fraud/enforcement/boise-nurse-practitioner-se... | November 7, 2024; U.S. Attorney's Office, Dist... |
| 2 | Former Traveling Nurse Pleads Guilty To Tamper... | November 7, 2024 | Criminal and Civil Actions | /fraud/enforcement/former-traveling-nurse-plea... | U.S. Attorney's Office, District of Massachusetts |
| 3 | Former Arlington Resident Sentenced To Prison ... | November 7, 2024 | Criminal and Civil Actions | /fraud/enforcement/former-arlington-resident-s... | U.S. Attorney's Office, Eastern District of Vi... |
| 4 | Paroled Felon Sentenced To Six Years For Fraud... | November 7, 2024 | Criminal and Civil Actions | /fraud/enforcement/paroled-felon-sentenced-to-... | U.S. Attorney's Office, Middle District of Flo... |
| 5 | Former Licensed Counselor Sentenced For Defrau... | November 6, 2024 | Criminal and Civil Actions | /fraud/enforcement/former-licensed-counselor-s... | U.S. Attorney's Office, Western District of Texas |
| 6 | Macomb County Doctor And Pharmacist Agree To P... | November 4, 2024 | Criminal and Civil Actions | /fraud/enforcement/macomb-county-doctor-and-ph... | U.S. Attorney's Office, Eastern District of Mi... |
| 7 | Rocky Hill Pharmacy And Its Owners Indicted Fo... | November 4, 2024 | Criminal and Civil Actions | /fraud/enforcement/rocky-hill-pharmacy-and-its... | U.S. Attorney's Office, Eastern District of Te... |
| 8 | North Texas Medical Center Pays $14.2 Million ... | November 4, 2024 | Criminal and Civil Actions | /fraud/enforcement/north-texas-medical-center-... | U.S. Attorney's Office, Northern District of T... |
| 9 | New England Doctor Pleads Guilty To Drug Distr... | November 4, 2024 | Criminal and Civil Actions | /fraud/enforcement/new-england-doctor-pleads-g... | U.S. Department of Justice |

# Step 2: Making the scraper dynamic

## 1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2) import time from datetime import datetime

function(int(month), int(year)): if year < 2013: print(“The date is invalid. Please retry with a year that is 2013 or later.”) return

base_url = ” link ” title = [] category = [] date = [] enforcement_agency = []

start_date = f”{year}-{month}-01” end_date = datetime.today()

page = 1 while True: url = f"{base_url}?page={page}"" for blank in df: scrape page using partnern section

```
start_date_month + 1, start_date_year+1 and <= end_date
time.sleep(1)
```

df = pd.DataFrame({ build dataframe}) df.to_csv("enforcement_actions_year_month")

return df

- b. Create Dynamic Scraper (PARTNER 2)

```
# Not printing the output of this as it was taking too long to knit. We call function later
        on
from bs4 import BeautifulSoup
import requests
from datetime import datetime
import time


def get_enforcement_actions(year, month):
    if year < 2013:
        print("The year is invalid. Please retry with a year that is 2013 or later.")
        return

    start_date = datetime(year, month, 1)
    today = datetime.today()

    titles, dates, links, categories = [], [], [], []

    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    page = 1

    while start_date <= today:
        if page == 1:
            url = base_url
        else:
            url = base_url + f"?page={page}"

        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'lxml')

        # getting titles and links
        h2_tags = soup.find_all('h2', class_="usa-card__heading")
        for h2 in h2_tags:
            a_tag = h2.find('a', href=True)
            if a_tag:
                links.append(a_tag['href'])
                titles.append(a_tag.get_text(strip=True))
            else:
                links.append("No link found")
                titles.append("No title found.")


        # getting dates
```

```
    header_tags = soup.find_all('header')
    for header in header_tags:
        div_tags = header.find_all('div')
        for div in div_tags:
            span_tag = div.find('span', class_="text-base-dark padding-right-105")
            if span_tag and span_tag.text.strip():
                dates.append(span_tag.text.strip())
            else:
                pass

    updated_dates = []
    for date in dates:
        try:
            format_change = datetime.strptime(date, "%B %d, %Y").strftime('%Y-%m-%d')
            updated_dates.append(format_change)
        except ValueError:
            pass

    filtered_dates = [date for date in updated_dates if date != "Invalid date"]


    # getting categories
    for header in header_tags:
        div_tags = header.find_all('div')
        for div in div_tags:
            ul_tags = div.find_all('ul')
            for ul in ul_tags:
                li_tags = ul.find_all('li', class_="display-inline-block usa-tag text-
 no-lowercase text-base-darkest bg-base-lightest margin-right-1")
                for li in li_tags:
                    categories.append(li.text.strip())


    categories = categories[:len(titles)]
    #print(f"Lengths of lists: titles={len(titles)}, categories={len(categories)},
     links={len(links)}, dates={len(filtered_dates)}") - commented out, was used to make
     sure function is working

    # create df of titles, category, link, and date
    main_df = pd.DataFrame({
        'Title' : titles,
        'Category': [", ".join(cat) for cat in categories],
        'Link': links,
        'Date': dates,
    })

    if page == 1:
        df = main_df
    else:
        df = pd.concat([df, main_df], ignore_index = True)

    page += 1
    time.sleep(0.2)
    # getting agency names from direct links
base_url = 'https://oig.hhs.gov'
```

```
    agency_info_list = []
    for link in df['Link']:
        agency_url = base_url + link
        # Fetch the page content
        response = requests.get(agency_url)

        # Parse the HTML content with Beautiful Soup
        soup = BeautifulSoup(response.text, 'html.parser')

        # Use the CSS selector path to find the element
        agency_info = soup.select_one('#main-content > div > div:nth-child(2) > article >
         div > ul > li:nth-child(2)')

        # Check if the element is found and print its text
        if agency_info:
            agency_info_list.append(agency_info.get_text(strip=True).replace("Agency:",
         "").strip())
        else:
            agency_info_list.append("Agency information not found.")


    df["Enforcement Agency"] = cleaned_agency_info_list

    #Save to CSV
    csv_name = f"enforcement_actions_{year}_{month}"
    df.to_csv(csv_name, index = False)
    return df

# Citation: Date scrapper was now including other aspects of the page with the same encoding
         so I created an if/else that continued to have errors. I entered the if/else into
         ChatGPT and it suggested try-except. Logic within is my own, just replaced if/else
         with try and except.

# Citation: Searched how to turn string date into datetime object with YYYY-MM-DD format
         and found stackoverflow thread that used the syntax used in this section for date
         conversion.
```

Citation: Received help from Ashirwad Wakade (mentioned in Google Form) to run function to extract dataframes for actions since January 2023 and for actions since January 2021, as our function took too long to run on our systems. Code for the function in Step 2 and 3 in this qmd is our own.

Creating Dataframe for January 2023 onwards

```
import pandas as pd
result_df_2023 = get_enforcement_actions(2023, 1)
```

```
# Importing as Excel as our function was taking too long when knitting
import pandas as pd
import altair as alt
import numpy as np
file_path = "C:/Users/prash/OneDrive/Documents/Python II/PSet
         5/enforcement_actions_2023_1.csv"

# Read the Excel file into a DataFrame
result_df_2023 = pd.read_csv(file_path)
```

```
length = len(result_df_2023)
print(length) # number of actions

last_row = result_df_2023.iloc[-1] # last row
print(last_row)

print(last_row['Title']) # full title
```

```
1534
Title       \nPodiatrist Pays $90,000 To Settle False Bill...
Date                                            January 3, 2023
Category                             Criminal and Civil Actions
Link    /fraud/enforcement/podiatrist-pays-90000-to-se...Agency
U.S. Attorney's Office, Southern District of T...Name: 1533,
dtype: object

Podiatrist Pays $90,000 To Settle False Billing Allegations
```

I got back 1534 enforcement actions in the dataframe. The earliest enforcement action is "Podiatrist Pays $90,000 To Settle False Billing Allegations" on January 3, 2023.

- c. Test Partner's Code (PARTNER 1)

Creating Dataframe for January 2021 onwards

```
import pandas as pd
result_df_2021 = get_enforcement_actions(2021, 1)
```

```
# Importing as Excel as our function was taking too long when knitting
import pandas as pd
import altair as alt
import numpy as np
file_path = "C:/Users/prash/OneDrive/Documents/Python II/PSet
         5/enforcement_actions_2021_1.csv"

# Read the Excel file into a DataFrame
result_df_2021 = pd.read_csv(file_path)
length = len(result_df_2021)
print(length) # number of actions

last_row = result_df_2021.iloc[-1] # last row
print(last_row)

print(last_row['Title']) # full title
```

```
3022
Title    \nThe United States And Tennessee Resolve Clai...Date
                                                January 4, 2021
Category                             Criminal and Civil Actions
Link    /fraud/enforcement/the-united-states-and-tenne...Agency
U.S. Attorney's Office, Middle District of Ten...Name: 3021,
dtype: object
```

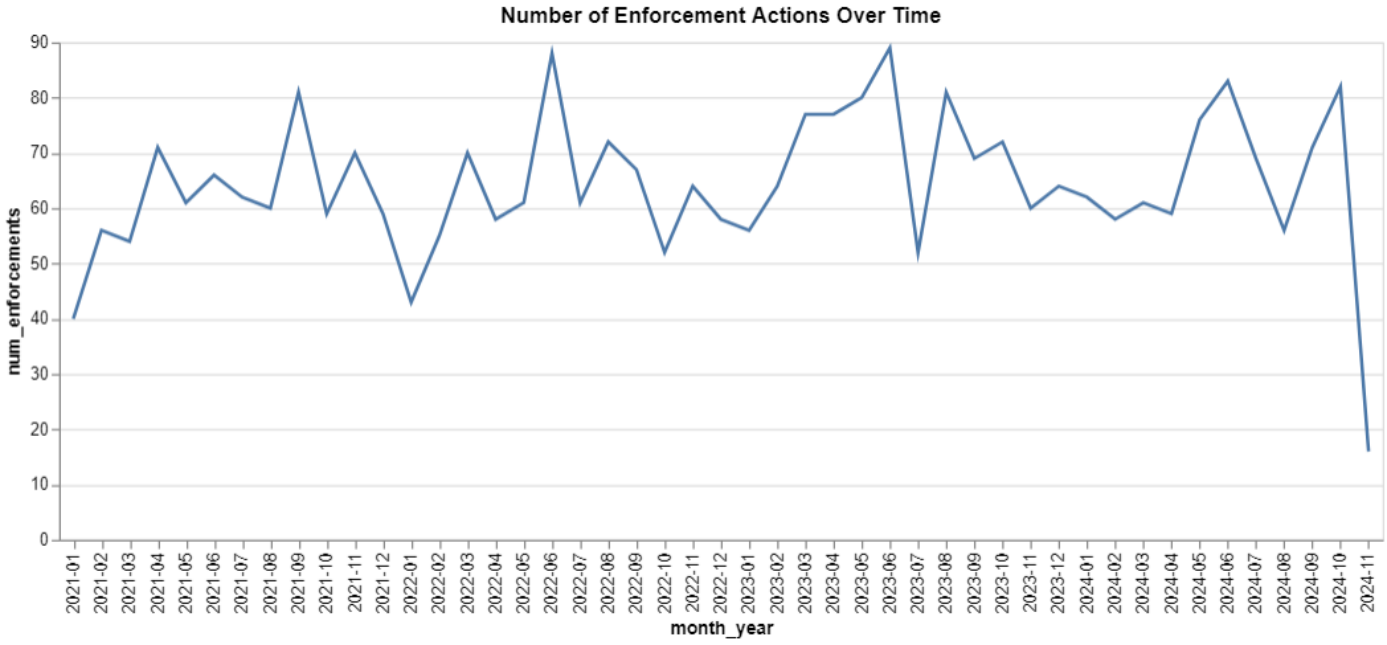
```
The United States And Tennessee Resolve Claims With Three Providers For False Claims Act
Liability Relating To ‘P-Stim’ Devices For A Total Of $1.72 Million
```

I got back 3022 enforcement actions in the dataframe. The earliest enforcement action is “The United States And Tennessee Resolve Claims With Three Providers For False Claims Act Liability Relating To ‘P-Stim’ Devices For A Total Of $1.72 Million” on January 4, 2021.

## Step 3: Plot data based on scraped data

### 1. Plot the number of enforcement actions over time (PARTNER 2)

```
import pandas as pd
import altair as alt
import numpy as np

file_path = "C:/Users/prash/OneDrive/Documents/Python II/PSet
        5/enforcement_actions_2021_1.csv"
result_df_2021 = pd.read_csv(file_path)
result_df_2021['Date'] = pd.to_datetime(result_df_2021['Date'], format='%B %d, %Y')
result_df_2021['month_year'] = result_df_2021['Date'].dt.to_period('M').astype(str)

enforcement_by_month =
        result_df_2021.groupby("month_year").size().reset_index(name="num_enforcements")
e_by_month_chart = alt.Chart(enforcement_by_month).mark_line().encode(
    x = "month_year:O",
    y = "num_enforcements:Q"
).properties(
    title = "Number of Enforcement Actions Over Time",
    width = 800
)
e_by_month_chart.display()
```



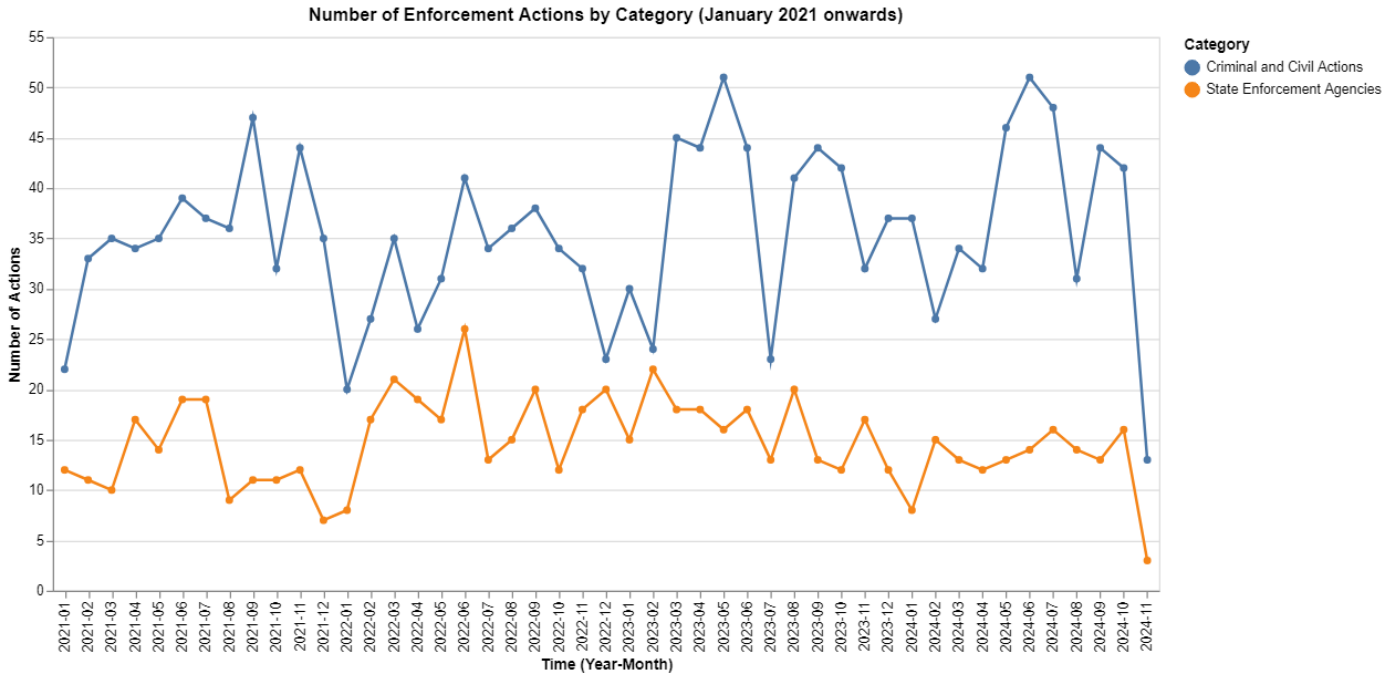
## 2. Plot the number of enforcement actions categorized: (PARTNER 1)

- based on “Criminal and Civil Actions” vs. “State Enforcement Agencies”

```
import pandas as pd
import altair as alt
import numpy as np

# Path to Excel file
file_path = "C:/Users/prash/OneDrive/Documents/Python II/PSet
         5/enforcement_actions_2021_1.csv"

# Read the Excel file into a DataFrame
result_df_2021 = pd.read_csv(file_path)

# Converting date from "January 1, 2021" to "01-01-2021"
result_df_2021['Date'] = pd.to_datetime(result_df_2021['Date'], format='%B %d, %Y')

# Extract month and year and combine them into a single column
result_df_2021['Month_Year'] = result_df_2021['Date'].dt.to_period('M').astype(str)

# Define the conditions
conditions = [
    result_df_2021['Category'] == 'State Enforcement Agencies',
    result_df_2021['Category'] == 'Criminal and Civil Actions'
]

# Define the corresponding choices
choices = [
    'State Enforcement Agencies',
    'Criminal and Civil Actions'
]

# Use np.select to create the new 'Category_New' column with three categories
result_df_2021['Category_New'] = np.select(conditions, choices, default='Other')

# Check the resulting DataFrame
result_df_2021


# Dataframe grouping by Month_Year and Category to summarize Number of Enforcement Actions
count_2021_category = result_df_2021.groupby(['Category_New', 'Month_Year']).agg(Count =
         ('Title', 'count')).reset_index()

# Filter out the "Other" category
filtered_df = count_2021_category[count_2021_category['Category_New'] != 'Other']

# Create the Altair chart
line_chart = alt.Chart(filtered_df).mark_line(point=True).encode(
    x=alt.X('Month_Year:O', title='Time (Year-Month)'),
    y=alt.Y('Count:Q', title='Number of Actions'),
    color=alt.Color('Category_New:N', title='Category')
).properties(
    title='Number of Enforcement Actions by Category (January 2021 onwards)',
    width=800,
    height=400
```

```
)
```

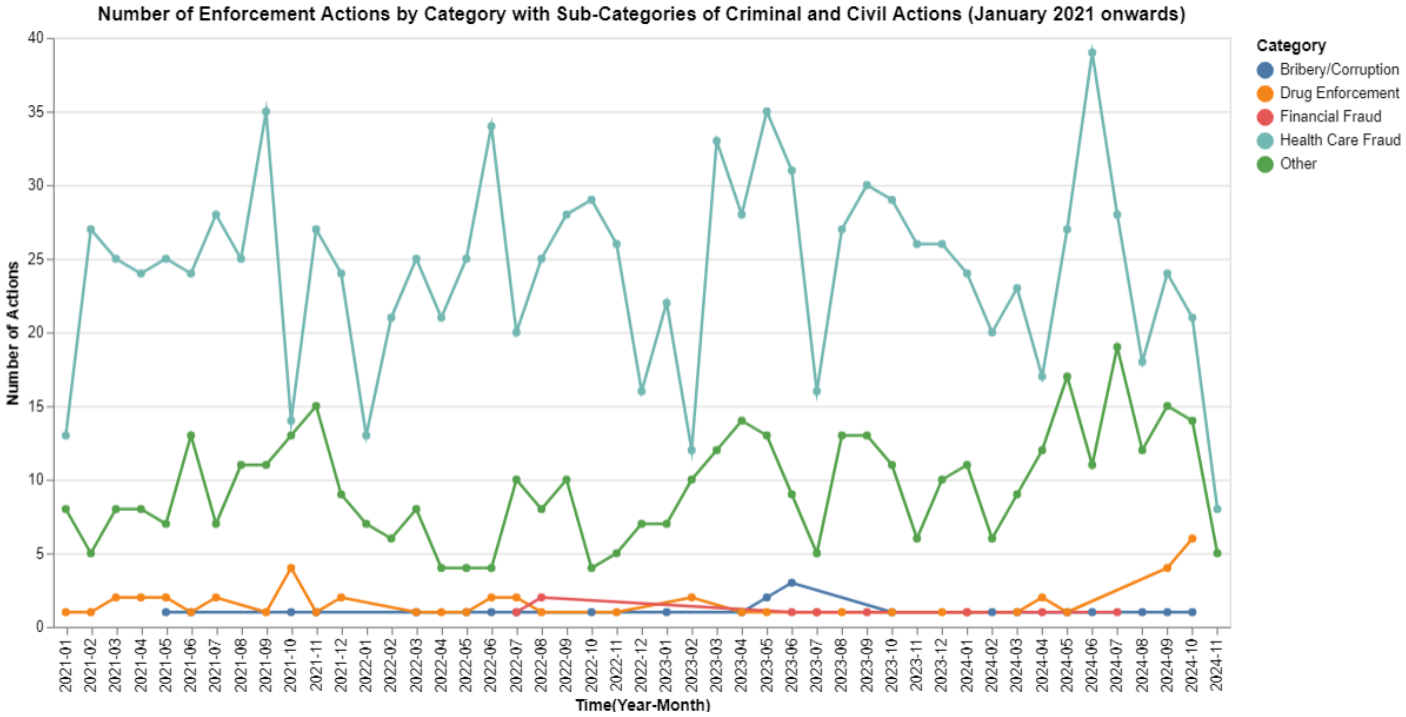
```
# Display the chart
line_chart.display()
```



- based on five topics

```
import pandas as pd
import altair as alt
import numpy as np

# Path to your Excel file
file_path = "C:/Users/prash/OneDrive/Documents/Python II/PSet
         5/enforcement_actions_2021_1.csv"

# Read the Excel file into a DataFrame
result_df_2021 = pd.read_csv(file_path)

# Converting date from "January 1, 2021" to "01-01-2021"
result_df_2021['Date'] = pd.to_datetime(result_df_2021['Date'], format='%B %d, %Y')

# Extract month and year and combine them into a single column
result_df_2021['Month_Year'] = result_df_2021['Date'].dt.to_period('M').astype(str)

# Define the conditions
conditions = [
    result_df_2021['Category'] == 'State Enforcement Agencies',
    result_df_2021['Category'] == 'Criminal and Civil Actions'
]

# Define the corresponding choices
choices = [
    'State Enforcement Agencies',
    'Criminal and Civil Actions'
]

# Use np.select to create the new 'Category_New' column with three categories
result_df_2021['Category_New'] = np.select(conditions, choices, default='Other')

# Filtering for only ['Category_New'] == "Criminal and Civil Actions", so that we can
         further classify
```

```
df_2021_civil_mask = result_df_2021['Category_New'] == "Criminal and Civil Actions"
df_2021_civil = result_df_2021[df_2021_civil_mask] # Queried to help correct it

# Sub-categorizing "Criminal and Civil Actions" into “Health Care Fraud”, “Financial Fraud”,
        “Drug Enforcement”, “Bribery/Corruption”, and “Other” - new colum names "Topic"
topics = { 'Health Care Fraud': ['health', 'care', 'pharmacist', 'doctor', 'medic'],
        'Financial Fraud': ['financial', 'bank', 'scam'], 'Drug Enforcement': ['drug',
        'raid', 'narcotics'], 'Bribery/Corruption': ['bribe', 'corruption', 'charges',
        'dollars'], 'Other': [] }

# Function to assign a topic to each action - queried in BingChat on how to assign a topic
        to each enforcement action
def assign_topic(title):
    for topic, keywords in topics.items():
        if any(keyword in title.lower() for keyword in keywords):
            return topic
    return 'Other'

# Using function on "Criminal and Civil Actions"
df_2021_civil['Topic'] = df_2021_civil.apply(lambda row: assign_topic(row['Title']), axis=1)

# Filtering for only ['Category_New'] == "State Enforcement Agencies" (not "Criminal and
        Civil Actions")
df_2021_state_mask = result_df_2021['Category_New'] == "State Enforcement Agencies"
df_2021_state = result_df_2021[df_2021_state_mask]
df_2021_state['Topic'] = df_2021_state['Category_New']

df_2021_other_mask = result_df_2021['Category_New'] == "Other"
df_2021_other = result_df_2021[df_2021_other_mask]
df_2021_other['Topic'] = "Misc non-Criminal and Civil Actions"

# Only data for Criminal and Civil Actions
combined_df_2021 = df_2021_civil
combined_df_2021['Topic'].unique()

# Dataframe grouping by Month_Year and Topic to summarize Number of Enforcement Actions
summary_df_2021 = combined_df_2021.groupby(['Topic', 'Month_Year']).agg(Count =('Title',
        'count')).reset_index()

# Create the Altair chart
line_chart_new = alt.Chart(summary_df_2021).mark_line(point=True).encode(
    x=alt.X('Month_Year:O',  title='Time(Year-Month)'),
    y=alt.Y('Count:Q', title='Number of Actions'),
    color=alt.Color('Topic:N', title='Category')
).properties(
    title='Number of Enforcement Actions by Category with Sub-Categories of Criminal and
        Civil Actions (January 2021 onwards)',
    width=800,
    height=400
)

# Display the chart
line_chart_new.display()
```

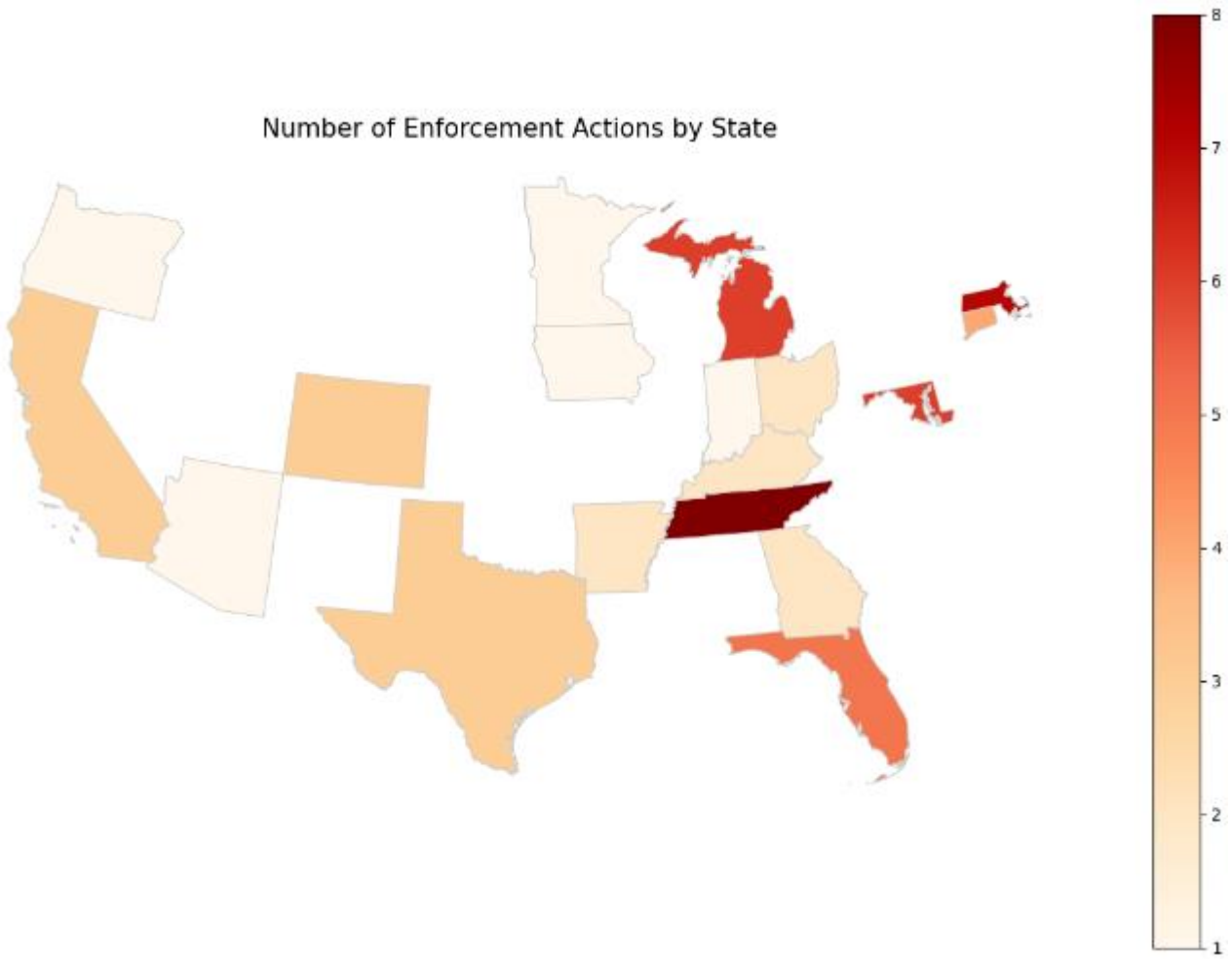

Number of Enforcement Actions by Category with Sub-Categories of Criminal and Civil Actions (January 2021 onwards)
Number of Actions
Time(Year-Month)
Category
Bribery/Corruption
Drug Enforcement
Financial Fraud
Health Care Fraud
Other

## Step 4: Create maps of enforcement activity

### 1. Map by State (PARTNER 1)

```
import pandas as pd
import warnings
import geopandas as gdp
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")

zip_filepath = "C:/Users/prash/Downloads/cb_2018_us_state_500k
        (1)/cb_2018_us_state_500k.shp"
zip_file = gdp.read_file(zip_filepath)
zip_file # NAME just has name of the state

# Path to your Excel file
file_path = "C:/Users/prash/OneDrive/Documents/Python II/PSet
        5/enforcement_actions_2021_1.csv"

# Read the Excel file into a DataFrame
combined_df_2021 = pd.read_csv(file_path)

# Create new column called name from Agency that just has the name of the state ("NAME")
combined_df_2021['NAME'] = combined_df_2021['Agency'].apply(lambda x: x.split('State of ')
        [1].split(' ')[0] if 'State of' in x else None)

combined_df_2021 = combined_df_2021.dropna(subset=['NAME'])

# Grouping Number of Actions by State ("NAME")
actions_by_state = combined_df_2021.groupby(['NAME']).agg(Count =('Title',
        'count')).reset_index()
actions_by_state.columns = ["NAME", "Count"]

# Merging dataframes
actions_by_state_merge = zip_file.merge(actions_by_state, left_on = "NAME", right_on =
        "NAME", how = "left")

# Making sure encoding is right
actions_by_state_merge = actions_by_state_merge.to_crs("EPSG:5070")
```

```
# Plotting Choropleth of hospitals by count of actions in each state
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
actions_by_state_merge.plot(column='Count', cmap='OrRd', linewidth=0.8, ax=ax,
         edgecolor='0.8', legend=True) # color scheme and formatting aided by BingChat
ax.set_title('Number of Enforcement Actions by State', fontsize=15)
ax.set_axis_off()
plt.show()
```



## 2. Map by District (PARTNER 2)

```
# Was not able to get it to work in time so did eval: false
import geopandas as gpd
file_path = "/Users/charismalambert/Documents/GitHub/problem-set-5-cl-ps"
districts_path = os.path.join(base_path,
"US_Attorney_Districts_Shapefile_simplified_20241109.csv")
districts = gpd.read_file(districts_path)

df_2021_path = os.path.join(base_path, "enforcement_actions_2021_1.csv")
df_2021 = pd.read_csv(df_2021_path)

# change column name to District
districts.rename(columns={"Judicial District": "District"}, inplace=True)
df_2021.rename(columns={"Agency": "District"}, inplace=True)
print(districts.columns)



# only keep district/agency name with District of in it
df_2021["District"] = df_2021["District"].str.extract(r"(District of. *)")
df_2021["District"] = df_2021["District"].str.strip()

# merge dataframes
merged_district_df = districts.merge(df_2021, how ="left", on = "District")

# update date to datetime object and group by district to count num_enforcements
enforcement_by_district =
merged_district_df.groupby("District").size().reset_index(name="num_enforcements")

district_enforcement_count = districts.merge(enforcement_by_district, how= "left", on
="District" )
print(district_enforcement_count.columns)

# convert to Geo dataframe
district_enforcement_count = gpd.GeoDataFrame(district_enforcement_count, geometry =
"the_geom")
district_enforcement_count = district_enforcement_count[["District", "num_enforcements",
"the_geom"]]
district_enforcement_count.set_crs("EPSG:4326", input = True)

# plot choropleth map
district_enforcement_count.plot(column = "num_enforcements", legend = True).set_axis_off()

#Citation: ChatGPT query of how to get certain parts of a string to be identified, it gave
back the use of regex, r"()" for finding District of within agency names inside of the df.
```